Probabilistic AI
Lecture 1: Introduction to variational inference and the ELBO

Helge Langseth*

Jan. 2021

*Slides made together with Thomas D. Nielsen, Aalborg University.
**Desiderata**

We seek to build models that:

- Reflect human understanding of a domain with transparent and explicit modelling assumptions.
- Sound semantics – both wrt. modelling language and interpretation of the generated results.
- Ability to capture fine structure in data
  - … yet robust towards noisy inputs, out-of-distribution queries, and adversarial attacks.
- Efficient inference algorithms
  - … giving results that are useful for making decisions under uncertainty.
- Supported by a useful “programming language” for simple(-ish) implementation.
Is a Deep Neural Network the solution?

Limits on the scope of deep learning*


- . . . is data hungry
- . . . is not sufficiently transparent
- . . . has not been well integrated with prior knowledge
- . . . presumes a largely stable world, in ways that may be problematic
- . . . works well as an approximation, but its answers often cannot be fully trusted

Is a *Deep Neural Network* the solution?

Limits on the scope of deep learning*

- . . . is data hungry
- . . . is not sufficiently transparent
- . . . has not been well integrated with prior knowledge
- . . . presumes a largely stable world, in ways that may be problematic
- . . . works well as an approximation, but *its answers often cannot be fully trusted*


Probabilistic AI = Deep Learning + Probabilistic thinking

A marriage of probabilistic thinking and deep learning is a framework that . . .
- . . . allows explicit modelling.
- . . . has a sound probabilistic foundation.
- . . . balances expert knowledge and information from data.
- . . . avoids restrictive assumptions about modelling families.
- . . . supports efficient inference.

**Probabilistic AI: A step towards **trustworthy **AI**
Plan for these lectures

Day 1: Introduction to variational inference and the ELBO
Dive into the mathematical details of Probabilistic AI, understand the foundation, and investigate the effects of some of the “shortcuts” being made.

- **Approximate inference** via the KL divergence, a.k.a. **Variational Bayes**
- The **mean-field** approach to Variational Bayes
- **Black Box variational inference**

Day 2: Disentanglement in the variational auto encoder
Devise flexible models for representation learning, and consider their transparency.

- **Variational Auto Encoders**
- **Disentanglement: What, why, how?**
- **Probabilistic Programming Languages**
Extended versions of these lectures (and more) are presented at the Nordic Probabilistic AI School, available at https://probabilistic.ai.

The 2023 Probabilistic AI School is available on Youtube, with accompanying source code on Github.
PGM Refresher
Each node is a random variable
- Edges indicate “influence” (Math-def: Graph encodes cond.indep. statements)
- For each variable $Y_k$, we must define $p(y_k \mid \text{pa}(y_k))$.
- The full model is defined as $p(y) = p(y_1, \ldots, y_n) = \prod_{i=1}^{n} p(y_i \mid \text{pa}(y_i))$.
- Markov properties $\Leftrightarrow$ Factorisation property.
Bayesian networks represent (high-dim) distributions over random variables.

- **Simple syntax:** Nodes, links, DAG, conditional distributions.
- **Clear semantics:** Nodes + links = Markov properties; Joint distribution.

**Inference:** Find \( p(z \mid x) \), where \( z \) are variables of interest, \( x \) are the observed variables, \( X \cup Z \subseteq Y \). (We will assume \( X \cup Z = Y \) throughout.)

**Note!** Evaluating \( p(y) = p(z, x) \) is simple: just use the definition of the model.

Evaluating \( p(x) \) for \( X \not\subseteq Y \) (and thus \( p(z \mid x) \)) is in general NP hard:

- **Exact inference:** Clever methods are available in some cases.
- **Approximate inference by sampling:** Markov Chain Monte Carlo is a common approximate solution.
- **Other approximate techniques:** This will be our approach!

**PGMs in these lectures**

Today we will look at the general inference problem, i.e., approximating \( p(z \mid x) \).

With that in place, we are ready to consider how to use PGMs for cool stuff tomorrow.
Variational Bayes: Approximate inference by optimization
The general goal is to somehow approximate $p(z \mid x)$ without too costly computational operations.

We will call the approximation $q(\cdot)$, hence hopefully “$q(z \mid x) \approx p(z \mid x)$”.

Often the conditioning part is dropped in $q(\cdot)$, hence $q(z)$ is a short-hand for $q(z \mid x)$.

**Formalization of approximate inference:**

Given a family of tractable distributions $Q$ and a distance measure between distributions $\Delta$, choose

$$\hat{q}(z) = \arg \min_{q \in Q} \Delta(q(z); p(z \mid x)).$$

**Decisions to be made:**

1. How to define $\Delta$ so that we end up with a high-quality solution from $Q$.
   - How to work with $\Delta(q(z); p(z \mid x))$ when we don’t even know what $p(z \mid x)$ is.

2. How to define a family of distributions $Q$ that is both flexible enough to generate good approximations and restrictive enough to support efficient calculations?
We want the $q$ closest to $p$. 

Illustration from Pyro documentation (https://pyro.ai/examples/intro_long.html)
Distance measure

Desiderata

To use $\Delta$ to measure the distance from an object $f$ to an object $g$ it would be relevant to require that $\Delta$ has the following properties:

**Positivity:** $\Delta(f; g) \geq 0$ and $\Delta(f; g) = 0$ if and only if $f = g$.

**Symmetry:** $\Delta(f; g) = \Delta(g; f)$

**Triangle:** For objects $f$, $g$, and $h$ we have that $\Delta(f; g) \leq \Delta(f; h) + \Delta(h; g)$.

Standard choice when working with probability distributions

It has become standard to choose the Kullback-Leibler divergence as the distance measure, where

$$KL(f \parallel g) = \mathbb{E}_{Z \sim f} \left[ \log \left( \frac{f(Z)}{g(Z)} \right) \right] = \int_z f(z) \log \left( \frac{f(z)}{g(z)} \right) dz.$$ 

Notice that while $KL(f \parallel g)$ obeys the positivity criterion, it satisfies neither symmetry nor the triangle inequality. It is thus not a proper distance measure.
## Two alternative KL definitions

### Moment-projection
- Minimizes \( \text{KL}(p||q) = -\mathbb{E}_{z \sim p}[\log q(z)] - \mathcal{H}_p \).
- Preference given to \( q \) that has:
  1. High \( q \)-probability allocated to \( p \)-probable regions.
  2. \( q(z) > 0 \) in any region where \( p \) is non-negligible.
  3. No explicit focus of entropy

### Information-projection
- Minimizes \( \text{KL}(q||p) = -\mathbb{E}_{z \sim q}[\log p(z)] - \mathcal{H}_q \).
- Preference given to \( q \) that has:
  1. High \( q \)-probability allocated to \( p \)-probable regions.
  2. Very small \( q \)-probability given to any region where \( p \) is small.
  3. High entropy ("large variance")

### Cheat-sheet:

- **KL-divergence:** \( \text{KL}(f||g) = \int_z f(z) \log \left( \frac{f(z)}{g(z)} \right) dz = \mathbb{E}_f \left[ \log \left( \frac{f(z)}{g(z)} \right) \right] \).
- **Entropy:** \( \mathcal{H}_f = -\int_z f(z) \log f(z) dz = -\mathbb{E}_f \left[ \log f(z) \right] \).
- **Intuition:** Cheat a bit (measure-zero, convergence rates, etc.) and think "If \( g(z_0) \approx 0 \), then \( -\mathbb{E}_{z \sim f} [\log g(z)] \) becomes ‘huge’ unless \( f(z_0) \approx 0 \)."
Moment and Information projection – main difference

Example: Approximating a Mix-of-Gaussians by a single Gaussian

- Moment projection – optimizing $\text{KL}(p || q)$ – has slightly larger variance.
- Similar mean values, but Information projection – optimizing $\text{KL}(q || p)$ – focuses mainly on the most prominent mode.
Moment and Information projection – main difference

Example: Approximating a Mix-of-Gaussians by a single Gaussian

- **Moment projection** – optimizing $\text{KL}(p\|q)$ – has slightly larger variance.

- Similar mean values, but **Information projection** – optimizing $\text{KL}(q\|p)$ – focuses mainly on the most prominent mode.

- **M-projection** is **zero-avoiding**, while **l-projection** is **zero-forcing**.
Variational Bayes w/ Mean Field
Variational Bayes relies on **information projections**, i.e., approximates $p(z \mid x)$ by

$$
\hat{q}(z) = \arg \min_{q \in \mathcal{Q}} \text{KL} (q(z)||p(z \mid x))
$$

- **Positives:**
  - Very efficient inference when combined with cleverly chosen $\mathcal{Q}$.
  - Clever interpretation when used for (Bayesian) learning.

- **Negatives:**
  - As we have seen, this may result in *zero-forcing* behaviour.
    - The typical choice of $\mathcal{Q}$ can make this issue even more prominent.
Notice how we can rearrange the KL divergence as follows:

\[
\text{KL}(q(z)||p(z|x)) = \mathbb{E}_{z \sim q}\left[ \log \frac{q(z)}{p(z|x)} \right] = \mathbb{E}_{z \sim q}\left[ \log \frac{q(z) \cdot p(x)}{p(z|x) \cdot p(x)} \right] \\
= \log p(x) - \mathbb{E}_{z \sim q}\left[ \log \frac{q(z)}{p(z, x)} \right] = \log p(x) - \mathcal{L}(q)
\]

The Evidence Lower Bound (ELBO) is

\[
\mathcal{L}(q) = -\mathbb{E}_q \left[ \log \frac{q(z)}{p(z, x)} \right] = \mathbb{E}_q \left[ \log \frac{p(z, x)}{q(z)} \right].
\]
Notice how we can rearrange the KL divergence as follows:

\[
\text{KL } (q(\mathbf{z}) \mid \mid p(\mathbf{z} \mid \mathbf{x})) = \mathbb{E}_{\mathbf{z} \sim q_\lambda} \left[ \log \frac{q(\mathbf{z})}{p(\mathbf{z} \mid \mathbf{x})} \right] = \mathbb{E}_{\mathbf{z} \sim q_\lambda} \left[ \log \frac{q(\mathbf{z}) \cdot p(\mathbf{x})}{p(\mathbf{z} \mid \mathbf{x}) \cdot p(\mathbf{x})} \right] \\
= \log p(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim q_\lambda} \left[ \log \frac{q(\mathbf{z})}{p(\mathbf{z}, \mathbf{x})} \right] = \log p(\mathbf{x}) - \mathcal{L}(q)
\]

The Evidence Lower Bound (ELBO) is

\[
\mathcal{L}(q) = -\mathbb{E}_q \left[ \log \frac{q(\mathbf{z})}{p(\mathbf{z}, \mathbf{x})} \right] = \mathbb{E}_q \left[ \log \frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z})} \right].
\]

**VB focuses on ELBO:**

\[
\log p(\mathbf{x}) = \mathcal{L}(q) + \text{KL } (q(\mathbf{z}) \mid \mid p(\mathbf{z} \mid \mathbf{x}))
\]

Since \(\log p(\mathbf{x})\) is constant with respect to the distribution \(q\) it follows:

- We can minimize \(\text{KL } (q(\mathbf{z}) \mid \mid p(\mathbf{z} \mid \mathbf{x}))\) by maximizing \(\mathcal{L}(q)\).
- This is **computationally simpler** because it uses \(p(\mathbf{z}, \mathbf{x})\) and not \(p(\mathbf{z} \mid \mathbf{x})\).
- \(\mathcal{L}(q)\) is a **lower bound** of \(\log p(\mathbf{x})\) because \(\text{KL } (q(\mathbf{z}) \mid \mid p(\mathbf{z} \mid \mathbf{x})) \geq 0\).

\(\Rightarrow\) During inference, we will look for \(\hat{q}(\mathbf{z}) = \text{arg max}_{q \in \mathcal{Q}} \mathcal{L}(q)\).
ELBO: Evidence Lower-BOund

Notice how we can rearrange the KL divergence as follows:

\[
\begin{align*}
\text{KL} \left( q(z) \mid \mid p(z \mid x) \right) &= \mathbb{E}_{z \sim q_{\lambda}} \left[ \log \frac{q(z)}{p(z \mid x)} \right] = \mathbb{E}_{z \sim q_{\lambda}} \left[ \log \frac{q(z) \cdot p(x)}{p(z \mid x) \cdot p(x)} \right] \\
&= \log p(x) - \mathbb{E}_{z \sim q_{\lambda}} \left[ \log \frac{q(z)}{p(z, x)} \right] = \log p(x) - \mathcal{L}(q)
\end{align*}
\]

The Evidence Lower Bound (ELBO) is

\[
\mathcal{L}(q) = -\mathbb{E}_{q} \left[ \log \frac{q(z)}{p(z, x)} \right] = \mathbb{E}_{q} \left[ \log \frac{p(z, x)}{q(z)} \right].
\]

**Summary:**

- We started out looking for the \( q \in \mathcal{Q} \) closest to \( p(z \mid x) \) in terms of \( \text{KL} \left( q(z) \mid \mid p(z \mid x) \right) \).

- An apparent problem is that we do not know what \( p(z \mid x) \) is, hence cannot calculate that distance.

- Still, we can find the optimal approximation by maximizing \( \mathcal{L}(q) \):

\[
\arg \max_{q \in \mathcal{Q}} \mathcal{L}(q) = \arg \min_{q \in \mathcal{Q}} \text{KL} \left( q(z) \mid \mid p(z \mid x) \right)
\]
What we have . . .

We now have the first building-block of the approximation:

\[ \Delta(q; p) = KL(q(z) || p(z | x)), \]

and avoided the issue with \( p(z | x) \) by focusing on \( \mathcal{L}(q) \).

We still need the set \( Q \):

Very often you will see the **mean field assumption**, which states that \( Q \) consists of all distributions that **factorizes** according to the equation

\[ q(z) = \prod_i q_i(z_i) \]

**Note!** This may seem like a very restrictive set. However, we can choose any \( q(z) \in Q \), and this is how the magic (\( \sim \) “absorbing information from \( x \)” ) happens.
Simple example
Bayesian regression model:

\[ Y_i \mid \{w_1, w_2, x_i\} = w_1 + w_2 x_i + \epsilon_i. \]

Notation: Write \( x_i \) for \([1, x_i]^T\). Then

\[ Y_i \mid \{w, x_i\} = w^T x_i + \epsilon_i \]
\[ \epsilon \sim N(0, 1/\gamma); \gamma \text{ known} \]
\[ w \sim N(\mu_0 = 0, \Sigma_0 = I_{2 \times 2}) \]

Data generated using \( w = [1, .5]^T \).
Simple example

Bayesian regression model:

\[ Y_i \mid \{w_1, w_2, x_i\} = w_1 + w_2 x_i + \epsilon_i. \]

Notation: Write \( x_i \) for \([1, x_i]^T\). Then

\[ Y_i \mid \{w, x_i\} = w^T x_i + \epsilon_i \]

\[ \epsilon \sim N(0, 1/\gamma); \gamma \text{ known} \]

\[ w \sim N(\mu_0 = 0, \Sigma_0 = I_{2 \times 2}) \]

Exact Bayesian solution:

\[ w \mid \{x, y, \gamma, \mu_0 = 0, \Sigma_0 = I_d\} \sim N(\gamma(I_d + \gamma X^T X)^{-1} X^T y, (I_d + \gamma X^T X)^{-1}) . \]
Simple example

Bayesian regression model:

\[ Y_i \mid \{ w_1, w_2, x_i \} = w_1 + w_2 x_i + \epsilon_i. \]

**Notation:** Write \( x_i \) for \([1, x_i]^T\). Then

\[ Y_i \mid \{ w, x_i \} = w^T x_i + \epsilon_i \]

\[ \epsilon \sim \mathcal{N}(0, 1/\gamma); \gamma \text{ known} \]

\[ w \sim \mathcal{N}(\mu_0 = 0, \Sigma_0 = I_{2 \times 2}) \]

Exact Bayesian solution:

\[ w \mid \{ x, y, \gamma, \mu_0 = 0, \Sigma_0 = I_d \} \sim \mathcal{N} \left( \gamma(I_d + \gamma X^T X)^{-1} X^T y, (I_d + \gamma X^T X)^{-1} \right) . \]

Variational Bayesian solution w/ Mean Field:

Iterative approach; assumes factorized posterior (that is, a Gaussian with diagonal covariance matrix).
Important observations from the example

- The **VB-MF** solution approximates the mean of the **true posterior** well.
  - Not **always** the case – depends on the problem.
- **VB-MF** totally **disregards correlation** between variables.
- **VB-MF** **under-estimates the uncertainty** of the **true posterior**.
  - Evident for the full joint, as well as for each marginal.
  - In this example, the **underestimation of each marginal variance** is by a factor $\sim 2.7$. 
Wrapping it all up: The VB algorithm under MF

Setup:

- We have observed $X = x$, and have access to the full joint $p(z, x)$.
- The posterior approximation is assumed to factorize according to the mean-field assumption, and we use the KL ($q(z) || p(z | x)$) as our objective.
- We posit a variational family of distributions $q_j(\cdot | \lambda_j)$, i.e., we choose the distributional form, while wanting to optimize the parameterization $\lambda_j$.
- The optimal $\lambda_j$ will depend on $x$ – in fact $\lambda_j$ encodes all the information about the other variables in the domain that $Z_j$ is “aware of”.

Algorithm:

Repeat until negligible improvement in terms of $\mathcal{L}(q)$:

1. For each $j$:
   - Somehow choose $\lambda_j$ to maximize $\mathcal{L}(q)$, typically based on $\{\lambda_i\}_{i \neq j}$. This can sometimes be done analytically, but today we will use a more general approach.
2. Calculate the new $\mathcal{L}(q)$. 

Probabilistic AI – Lecture 1

Variational Bayes w/ Mean Field
Stochastic Gradient Ascent
Gradient ascent algorithm for maximizing a function $f(\mathbf{\lambda})$:

1. Initialize $\mathbf{\lambda}^{(0)}$ randomly.
2. For $t = 1, \ldots$ :

   \[ \mathbf{\lambda}^{(t)} \leftarrow \mathbf{\lambda}^{(t-1)} + \rho \cdot \nabla_{\mathbf{\lambda}} f\left(\mathbf{\lambda}^{(t-1)}\right) \]

$\mathbf{\lambda}^{(t)}$ converges to a (local) optimum of $f(\cdot)$ if:

- $f$ is “sufficiently nice”;
- The learning-rate $\rho$ is “sufficiently small”.

Why do we talk about this?

We want a way to optimize ELBO using gradient methods. If we can do inference as optimization it will play well with, e.g., deep learning frameworks.
Example: ML in a Gaussian model

Example: Maximum log likelihood in a Gaussian model

We have access to $N = 1000$ observations from a Gaussian distribution with unknown mean $\mu$ and precision $\tau$. Use $\lambda = [\mu, \tau]^T$.

$$f(\lambda) = \sum_{i=1}^{N} \log p(x_i | \lambda) = \frac{N}{2} \log \tau - \frac{N}{2} \log(2\pi) - \frac{\tau}{2} \sum_{i=1}^{N} (x_i - \mu)^2$$

$$\nabla_\lambda f(\lambda) = \begin{bmatrix}
-N\tau \mu + \tau \sum_{i=1}^{N} x_i \\
\frac{N}{2\tau} - \frac{1}{2} \sum_{i=1}^{N} (x_i - \mu)^2
\end{bmatrix}$$

Cost of calculation: $O(N)$
Stochastic gradient ascent algorithm for maximizing a function $f(\lambda)$:

If we have access to $g(\lambda)$ – an **unbiased estimate** of the gradient – it still works!

1. Initialize all variational parameters randomly to $\lambda^{(0)}$.
2. For $t = 1, \ldots$:

\[
\lambda^{(t)} \leftarrow \lambda^{(t-1)} + \rho_t \cdot g\left(\lambda^{(t-1)}\right)
\]

$\lambda_t$ converges to a (local) optimum of $f(\cdot)$ if:

- $f$ is “sufficiently nice”;
- $g(\lambda)$ is a random variable with $\mathbb{E}[g(\lambda)] = \nabla_\lambda f(\lambda)$ and finite variance.
- The learning-rates $\{\rho_t\}$ is a Robbins-Monro – sequence:
  - $\sum_t \rho_t = \infty$
  - $\sum_t \rho_t^2 < \infty$
Example: Maximum log likelihood in a Gaussian model

We consider the same maximum likelihood problem, but instead of the gradient based on the full sample, we only have a mini-batch of a single example $x_t$ at iteration $t$:

$$g(\lambda | x_t) = N \cdot \left[ \frac{-\tau \mu + \tau x_t}{2\tau} - \frac{1}{2} (x_t - \mu)^2 \right]$$

**Cost of calculation:** $O(1)$

Randomness in $g$ is a consequence of the random data selection process; multiplying by $N$ ensures that $\mathbb{E}[g(\lambda)] = \nabla_\lambda f(\lambda)$.
Black Box Variational Inference
Main idea: Cast inference as an optimization problem

Optimize the ELBO by stochastic gradient ascent over the parameters $\lambda$.

Algorithm: Maximize $\mathcal{L}(q) = \mathbb{E}_{q_{\lambda}} \left[ \log \frac{p_{\theta}(z, x)}{q_{\lambda}(z)} \right]$ by gradient ascent

- Initialization:
  - $t \leftarrow 0$;
  - $\hat{\lambda}_0 \leftarrow$ random initialization;
  - $\rho \leftarrow$ a Robbins-Monro sequence.

- Repeat until negligible improvement in terms of $\mathcal{L}(q)$:
  - $t \leftarrow t + 1$;
  - $\hat{\lambda}_t \leftarrow \hat{\lambda}_{t-1} + \rho_t \nabla_{\lambda} \mathcal{L}(q) |_{\hat{\lambda}_{t-1}}$;

Important issue:
Can we calculate $\nabla_{\lambda} \mathcal{L}(q)$ efficiently without adding new restrictive assumptions?
The algorithm requires that we can find

$$\nabla_\lambda \mathcal{L} (q) = \nabla_\lambda \mathbb{E}_{z \sim q_\lambda} \left[ \log \frac{p_\theta(z, x)}{q_\lambda(z \mid \lambda)} \right].$$

We can use these properties to simplify the equation:

1. $\nabla_\lambda (f(z, \lambda) \cdot g(z, \lambda)) = f(z, \lambda) \cdot \nabla_\lambda g(z, \lambda) + g(z, \lambda) \nabla_\lambda f(z, \lambda)$
2. $\nabla_\lambda f(z, \lambda) = f(z, \lambda) \nabla_\lambda \log f(z, \lambda)$
3. $\mathbb{E}_{q_\lambda} [\nabla_\lambda \log q_\lambda(z \mid \lambda)] = 0$ for a density function $q_\lambda(z \mid \lambda)$

Now it follows that

$$\nabla_\lambda \mathcal{L} (q) = \mathbb{E}_{z \sim q_\lambda} \left[ \log \frac{p_\theta(z, x)}{q_\lambda(z \mid \lambda)} \cdot \nabla_\lambda \log q_\lambda(z \mid \lambda) \right].$$
Calculating the gradient – Things to notice

\[ \nabla_{\lambda} \mathcal{L}(q) = \mathbb{E}_{z \sim q_{\lambda}} \left[ \log \frac{p_{\theta}(z, x)}{q_{\lambda}(z | \lambda)} \right] \cdot \nabla_{\lambda} \log q_{\lambda}(z | \lambda) \]
We still only need access to the joint distribution $p(\mathbf{z}, \mathbf{x})$ – not $p(\mathbf{z} | \mathbf{x})$.

\[ \nabla_\lambda \mathcal{L}(q) = \mathbb{E}_{\mathbf{z} \sim q_\lambda} \left[ \log \frac{p(\mathbf{z}, \mathbf{x})}{q_\lambda(\mathbf{z} | \lambda)} \right] \cdot \nabla_\lambda \log q_\lambda(\mathbf{z} | \lambda). \]

Calculating the gradient – Things to notice
Calculating the gradient – Things to notice

- We still only need access to the joint distribution \( p(\mathbf{z}, \mathbf{x}) \) – not \( p(\mathbf{z} \mid \mathbf{x}) \).

\[
\nabla_{\lambda} \mathcal{L}(q) = \mathbb{E}_{\mathbf{z} \sim q_\lambda} \log \frac{p(\mathbf{z}, \mathbf{x})}{q_\lambda(\mathbf{z} \mid \lambda)} \cdot \nabla_{\lambda} \log q_\lambda(\mathbf{z} \mid \lambda).
\]

- \( q_\lambda(\mathbf{z} \mid \lambda) \) factorizes under MF, s.t. we can optimize per variable: \( q_{\lambda_i}(z_i \mid \lambda_i) \).
Calculating the gradient – Things to notice

- We still only need access to the joint distribution \( p_\theta(z, x) \) – not \( p_\theta(z \mid x) \).

\[
\nabla_\lambda \mathcal{L}(q) = \mathbb{E}_{z \sim q_\lambda} \left[ \log \frac{p_\theta(z, x)}{q_\lambda(z \mid \lambda)} \right] \cdot \nabla_\lambda \log q_\lambda(z \mid \lambda).
\]

- \( q_\lambda(z \mid \lambda) \) factorizes under **MF**, s.t. we can optimize per variable: \( q_\lambda_i(z_i \mid \lambda_i) \).

- We must calculate \( \nabla_\lambda \log q(z_i \mid \lambda_i) \), which is also known as the “score function”. This depends on the distributional family of \( q(\cdot) \); can be precomputed for standard distributions and auto-diff’ed for more complex constructions.
Calculating the gradient – Things to notice

- We still only need access to the joint distribution $p_{\theta}(z, x)$ – not $p_{\theta}(z \mid x)$.

$$\nabla_{\lambda} \mathcal{L}(q) = \mathbb{E}_{z \sim q_{\lambda}} \left[ \log \frac{p_{\theta}(z, x)}{q_{\lambda}(z \mid \lambda)} \cdot \nabla_{\lambda} \log q_{\lambda}(z \mid \lambda) \right] .$$

- $q_{\lambda}(z \mid \lambda)$ factorizes under MF, s.t. we can optimize per variable: $q_{\lambda_i}(z_i \mid \lambda_i)$.

- We must calculate $\nabla_{\lambda_i} \log q(z_i \mid \lambda_i)$, which is also known as the “score function”. This depends on the distributional family of $q(\cdot)$; can be precomputed for standard distributions and auto-diff’ed for more complex constructions.

- The expectation will be approximated using a sample $\{z_1, \ldots, z_M\}$ generated from $q(z \mid \lambda)$. Hence we require that we can sample from $q_{\lambda_i}(\cdot)$. 

Probabilistic AI – Lecture 1  Black Box Variational Inference
Calculating the gradient – Things to notice

- We still only need access to the joint distribution $p_\theta(z, x)$ – not $p_\theta(z | x)$.

$$\nabla_\lambda \mathcal{L}(q) = \mathbb{E}_{z \sim q_\lambda} \left[ \log \frac{p_\theta(z, x)}{q_\lambda(z | \lambda)} \cdot \nabla_\lambda \log q_\lambda(z | \lambda) \right].$$

- $q_\lambda(z | \lambda)$ factorizes under MF, s.t. we can optimize per variable: $q_{\lambda i} (z_i | \lambda_i)$.
- We must calculate $\nabla_{\lambda_i} \log q (z_i | \lambda_i)$, which is also known as the “score function”. This depends on the distributional family of $q(\cdot)$; can be precomputed for standard distributions and auto-diff’ed for more complex constructions.
- The expectation will be approximated using a sample $\{z_1, \ldots, z_M\}$ generated from $q(z | \lambda)$. Hence we require that we can sample from $q_{\lambda i}(\cdot)$.

Calculating the gradient – in summary

We have observed the datapoint $x$, and our current estimate for $\lambda_i$ is $\hat{\lambda}_i$. Then

$$\nabla_{\lambda_i} \mathcal{L}(q) |_{\lambda = \hat{\lambda}_i} \approx \frac{1}{M} \sum_{j=1}^{M} \log \frac{p(z_j, x)}{q_{\lambda_i}(z_{i,j} | \hat{\lambda}_i)} \cdot \nabla_{\lambda_i} \log q_{\lambda_i}(z_{i,j} | \hat{\lambda}_i).$$

where $\{z_{i,1}, \ldots, z_{i,M}\}$ are samples from $q_{\lambda_i}(\cdot | \hat{\lambda}_i)$. 
Black Box Variational Inference

Black box variational inference is a **general purpose** approach for VI, that can maximize $\mathcal{L}(q)$ if we are able to . . .

- **sample** from $q_{\lambda_i}(z_i | x, \lambda_i)$;
- **calculate the “score function”** $\nabla_{\lambda_i} \log q_{\lambda_i}(z_i | x, \lambda_i)$.

Since $q_{\lambda_i}(z_i | x, \lambda_i)$ is under our control, this should be OK, e.g., by letting $q_{\lambda_i}(\cdot)$ be a standard distribution parameterized by a DNN (input $x$; weights $\lambda_i$).

Consequences

- Since probabilistic inference now is done by gradient methods, we can rely on **autodiff-tools** like Tensorflow and Pytorch to work with arbitrarily complex distributions.
- Probabilistic modelling can thus be **seamlessly integrated** with building-blocks from other machine learning approaches (like deep learning).
  - We can e.g. represent $q(\theta | D)$ via a DNN, and iteratively tune the DNN's weights while calculating the posterior (given the weights).
- We will see an example of this tomorrow, in the **Variational Auto Encoder**.